Lab 4: Correlation by hand, the gruesome details

Statistics for Social Scientists II

Bur, GJM

2024-09-30

1 Understanding correlation: working "by hand" in Stata

Good news, everyone! (Say that in Professor Frink's voice for added "hilarity"). We've already done most of the work necessary to complete this week's homework. So, I'm just going to reproduce some of the more ambitious questions from last week that are especially relevant to this week.

First, let me note the following. Prof. Gerber showed in class one of several ways of writing the correlation coefficient. Let's catalogue some of the ones I've shown so far…

1) The "cosine" formula, derived on the board last lab and in my 360 notes (see, e.g., here).

$$r = \hat{\rho} = \frac{\sum_{j=1}^{n}(x_j - \bar{x})(y_j - \bar{y})}{\sqrt{\sum_{j=1}^{n}(x_j - \bar{x})^2}\sqrt{\sum_{j=1}^{n}(y_j - \bar{y})^2}}$$

   You can think of this as the "cosine" formula because in this way of writing it, the numerator is the dot product of two vectors (where each coordinate of the vector is some x_j less the x-mean or some y_j less the y-mean) and the denominator is the product of the lengths of two vectors (notice that those sums are just the Pythagorean theorem!).

2. The "standardized covariance formula.

   We get this one by simply writing an $n-1$ underneath both numerator and denominator and then doing a bit of tedious but easy algebra.

   Here's how we get from the standardized covariance formula to the z-score formula.

$$
\begin{aligned}
r = \hat{\rho} &= \frac{\sum_{j=1}^{n}(x_j-\bar{x})(y_j-\bar{y})}{\sqrt{\sum_{j=1}^{n}(x_j-\bar{x})^2}\sqrt{\sum_{j=1}^{n}(y_j-\bar{y})^2}} \cdot \frac{n-1}{n-1} \\
&= \frac{\sum_{j=1}^{n}(x_j-\bar{x})(y_j-\bar{y})}{\sqrt{\sum_{j=1}^{n}(x_j-\bar{x})^2}\sqrt{\sum_{j=1}^{n}(y_j-\bar{y})^2}} \cdot \frac{\sqrt{n-1}\sqrt{n-1}}{n-1} \\
&= \frac{\sum_{j=1}^{n}(x_j-\bar{x})(y_j-\bar{y})}{n-1} \cdot \frac{1}{\frac{\sqrt{\sum_{j=1}^{n}(x_j-\bar{x})^2}}{\sqrt{n-1}}} \cdot \frac{1}{\frac{\sqrt{\sum_{j=1}^{n}(y_j-\bar{y})^2}}{\sqrt{n-1}}} \\
&= \frac{\sum_{j=1}^{n}(x_j-\bar{x})(y_j-\bar{y})}{n-1} \cdot \frac{1}{\sqrt{\frac{\sum_{j=1}^{n}(x_j-\bar{x})^2}{n-1}}} \cdot \frac{1}{\sqrt{\frac{\sum_{j=1}^{n}(y_j-\bar{y})^2}{n-1}}}. \\
&= \hat{\mathbb{COV}}[X,Y] \cdot \frac{1}{\hat{\sigma}_X\hat{\sigma}_Y}
\end{aligned}
$$

3. The "(near)-mean z-product" formula.

   We get this one by simply regrouping in the penultimate step above.

$$
\begin{aligned}
r &= \frac{\sum_{j=1}^{n}(x_j-\bar{x})(y_j-\bar{y})}{n-1} \cdot \frac{1}{\hat{\sigma}_X\hat{\sigma}_Y} \\
&= \sum_{j=1}^{n}\left\{\frac{x_j-\bar{x}}{\hat{\sigma}_X}\right\}\left\{\frac{y_j-\bar{y}}{\hat{\sigma}_Y}\right\} \cdot \frac{1}{n-1} \\
&= \frac{\sum_{j=1}^{n} z_{xj}z_{yj}}{n-1}
\end{aligned}
$$

There are actually at least 13 good ways to look at the correlation coefficient (see Rodgers and Nicewander). It is truly a beautiful formula. It's also the regression slope for standardized variables, among other things. The formula Prof. Gerber showed in Thursday's lecture is also a very useful one, but let me show you a somewhat simpler version of it that gets at the same ideas. **This version was historically sometimes known as the "computational" formula**; it essentially uses the "König-Huygens" trick for the variance and covariance. I'll re-derive that below.

First, note that…

$$
\begin{aligned}
\sum_{j=1}^{n}(y_j-\bar{y})^2 &= \sum_{j=1}^{n} y_j^2 - 2\sum_{j=1}^{n} y_j\bar{y} + \sum_{j=1}^{n}\bar{y}^2 \\
&= n\overline{y^2} - 2n\bar{y}^2 + n\bar{y}^2 \\
&= n(\overline{y^2} - \bar{y}^2)
\end{aligned}
$$

Then, note that by identical logic…

$$\begin{aligned}\sum_{j=1}^{n}(y_j-\bar{y})(x_j-\bar{x}) &= \sum_{j=1}^{n}x_jy_j-\sum_{j=1}^{n}y_j\bar{x}-\sum_{j=1}^{n}x_j\bar{y}+\sum_{j=1}^{n}\bar{x}\bar{y}\\ &= n\overline{xy}-2n\bar{x}\bar{y}+n\bar{x}\bar{y}\\ &= n(\overline{xy}-\bar{x}\bar{y})\end{aligned}$$

So, overall, if …

$$r=\hat{\rho}=\frac{\sum_{j=1}^{n}(x_j-\bar{x})(y_j-\bar{y})}{\sqrt{\sum_{j=1}^{n}(x_j-\bar{x})^2}\sqrt{\sum_{j=1}^{n}(y_j-\bar{y})^2}}$$

Then we can also simply find…

$$\begin{aligned}r &= \frac{n(\overline{xy}-\bar{x}\bar{y})}{\sqrt{n(\overline{x^2}-\bar{x}^2)}\sqrt{n(\overline{y^2}-\bar{y}^2)}}\\ &= \frac{\overline{xy}-\bar{x}\bar{y}}{\sqrt{(\overline{x^2}-\bar{x}^2)}\sqrt{(\overline{y^2}-\bar{y}^2)}}\end{aligned}$$

This is a simpler version of the formula that Prof. Gerber showed (and probably a little more common to see), but it comes to the same thing (if you want, you can show that his formula gives this one; see appendix for hints).

In words: you can find the correlation coefficient by making a product variable (multiply together two variables whose correlation you want to find) and then take their mean; then subtract that from the product of their individual means. That is *approximately* the sample covariance (it's a bit off because of the substitution of $n-1$ for n in the denominator of the *unconditional* sample mean squared error (AKA variance), but don't worry; since our standard deviation approximations are both divided by by a factor of $\sqrt{n-1}$, it will all cancel out!)

Then, approximate each variable's variance using the same trick: make two new variables that are each the square of the old variable. Then the sample mean of this new variable, say $\overline{y^2}$, less the square of the old variable's mean, say $\bar{y}^2$, is close to the old variable's variance (you would need to multiply this quantity by $\frac{n}{n-1}$ to get the exact number).

2 Exercises for this week

Your task for this lab is essentially to re-try some of those harder questions from last week's lab.

Here's one reproduced question from last time, with a slight twist: we'll now use the "computational formula".

2.1 Correlation by "hand" in Stata

1. Use the `auto` data-set and let `x` be `weight` and `y` be `mpg`. We'll generally try to predict the miles-per-gallon that a car gets from its weight.

   This week, I insist that you try to do this by hand in Stata.

   First, make a "product variable" called `prod` and find its mean. Then subtract from that the product of the individual variables' means; for large n, this is almost exactly equal to the covariance of the two variables. *When we divide this by the approximation for the variances, it will actually be exactly correct because whether you take the true mean* (the machine learning approach) *or the bias-corrected mean* (n-1, the statistical approach), **it will cancel out**.

   Then, find the correlation by dividing this covariance by the product of the variables' individual standard deviations. Again, do this by "hand": successively `summarize` the two variables, then for each variable make a new variable that is equal to the square of the old variable less its mean. The difference[1] of these two quantities will approximately be the variance of each variable, respectively.

   Here's some example code. ***Remember that lines involving locals must be run all at once to work. You do not have to do this with locals for homework if you are not comfortable***, but it is the fastest way (computationally) and the clearest way (because you just write down the formulae in their abstractness, not having to jot down specific numbers that will change with every problem). Of course, the fastest and best way is to do this in a loop.

   A couple more notes: I'm going to drop the <> delimiters. So, just know that you should input your own variables wherever you see x etc.—anything that's obviously a placeholder variable (or think of a clever way to avoid all that work…)

   ```
   drop if missing(x, y)
   * make product var
   gen prod = x*y
   sum prod
   local prodmean = r(mean)
   sum x
   local mean1 = r(mean)
   sum y
   local mean2 = r(mean)
   di `prodmean' - `mean1'*`mean2'
   corr x y, cov // checking work; if approximately true, store in local
   local cov = `prodmean' - `mean1'*`mean2'

   * make variance var
   sum x
   ```

[1] Last time, I think I wrote "mean of the new variable" here, but obviously I meant to write the difference of the two.

```
local mean1 = r(mean)
    // we already defined this, but just doing this so that we can run the code in chunks
gen xsq = x^2
sum xsq
local v_x = r(mean) - `mean1'^2
local sd1 = sqrt(`v_x')
di `sd1'
sum x // again, just checking our work

* repeat this for y

local r = `cov'/(`sd1'*`sd2')
di `r'
corr x y
assert round(`r', 0.000001) == round(r(C)[2,1], 0.000001)
    // Recall that "r(C)[row, column]" calls the element of the matrix
    // that we just printed with -corr-
    // Then, "round" just rounds the first argument to the decimal
    // place specified in the second argument.
    // Finally, -assert- should, somewhat counterintuitively, produce
    // *nothing* if the statement is true.
```

2.2 Regression “by hand” in Stata: $\widehat{\beta}_1$ from $\widehat{\sigma}_{X,Y}$ and $\widehat{\sigma}_X$

2. Find the regression slope and intercept for the regression of `mpg` on `weight`.[2] *Do this using the covariance and the variance of X.*

```
* no explicit code given here.
* use the fact that for simple regression with OLS
* we simply need s_{x,y}/s^2_x
* hint on checking work: you can get the coeffs from
* regression with ...
reg y x
matrix list e(b)
// note that to get the vector of coeffs,
// you must write -reg y x- to get in Stata's
// "working memory"
* and then access the coefficients with matrix notation
* e.g. to get the slope
di e(b)[1,1]
```

2.3 Regression “by hand” in Stata: $\widehat{\beta}_0$ from $\widehat{\beta}_1$ and $\bar{x}, \bar{y}$

3. Obtain the regression intercept using the formula (justified in the appendix) with $\widehat{\beta}_0 = \bar{y} - \widehat{\beta}_1\bar{x}$. **Note once again that since we approximated the covariance and the variance in the same way, we don’t actually need to worry about the $n - 1$ correction factor**.

[2] Thanks to Emily from SOC361fa24 at the UW-Madison for finding several minor typos!

```
* no code here; this should be a pretty easy extension of the above
* note that we have already defined all the relevant quantities
```

2.4 Regression "by hand" in Stata: R^2 from r

4. Obtain the model fit (proportion of variance explained), aka R^2 (see appendix for the fascinating reason, involving basic geometry, that the correlation coefficient for simple regression is also the proportion of variance explained).

```
* no code here; this should be a pretty easy extension of the above
* note that we have already defined all the relevant quantities
```

2.5 Regression "by hand" in Stata: $\hat{\sigma}_{\hat{\beta}_1}, \hat{\sigma}_{\hat{\beta}_0}$

5. Obtain the standard errors for each slope.[3] Recall that...

$$\hat{\sigma}^2_{\hat{\beta}_1} = \frac{\hat{\sigma}^2}{\sum_{j=1}^{n}(x_j - \bar{x})^2}$$

One way to do this for $\hat{\beta}_1$ is to make a variable with the predictions, and then make a new variable that subtracts that from the outcome, and then squares it. The mean of this variable is very close to the mean squared error; it is exact if you multiply the mean by $\frac{n}{n-2}$. Then, the den

```
gen yhat = x*`b1' + `b0' // this makes a variable containing predictions
gen squarederrors = (y - yhat)^2
    // this makes a variable which is squared errors
sum squarederrors
local mse_unadj = r(mean)
    // this is an unadjusted mse, divided by a df
    // that *standard* stats theory says is wrong
local mse = `mse_unadj' * r(N)/(r(N)-2)
    // now we adjust by the standard df
di `mse'
reg y x
assert round(`mse', 0.01) == round(e(rmse)^2, 0.01)
    // we check that our mse is the same as Stata
sum x, d
local SE_b1 = sqrt(`mse'/((r(N)-1)*r(Var)))
    // now we get our SE by taking the root of the
    // mse by the sum of squares of the Xs. Do you
    // see why the code does that?
di `SE_b1'
* Accessing the standard errors from a matrix of results
* is a bit trickier, but here is how you do it
```

[3] FYI: $\hat{\beta}_0$ is called a slope because technically it is a slope on a vector of 1s that any software automatically adds when estimating a regression unless you tell it not to include an intercept.

```
reg y x
assert round(`SE_b1', 0.000001) == round(r(table)[2,1], 0.000001)
* as always, write
matrix list r(table)
* to see what it contains
```

Now, how do we do this for $\hat{\beta}_0$?

Recall that its *sampling variance* (the square of the SE) is a bit cleaner to write. It is …

$$\hat{\sigma}^2_{\hat{\beta}_0} = \hat{\sigma}^2 \cdot \frac{\overline{x^2}}{\sum_{j=1}^{n}(x_j - \bar{x})^2}$$

```
sum x, d
local SS_X = (r(N)-1)*r(Var) // we'll need
sum xsq
local SE_b0 = sqrt(`mse' * r(mean)/`SS_X')
di `SE_b0'
reg y x
assert round(`SE_b0', 0.000001) == round(r(table)[2,2], 0.000001)
```

By the way, note that R^2 is *also* equal to 1 less the sum of squared errors over the outcome sum of squares:

$$\begin{aligned} R^2 &= 1 - \frac{SS_e}{SS_T} \\ &= 1 - \frac{\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}{\sum_{j=1}^{n}(y_j - \bar{y}_j)^2} \end{aligned}$$

So…

```
sum squarederrors
local sse = r(N)*r(mean)
sum y, d
local sst = (r(N)-1)*r(Var)
local r2 = 1 - `sse'/`sst'
di `r2'
corr y x
assert round(`r2', 0.000001) == round((r(C)[2,1])^2, 0.000001)
```

I leave the t-statistics up to you!

2.6 Scatterplots in Stata

6. There is no clever way to do this by "hand" unless you want to get *really* into programming (I'm not even that into programming). However, I ***did*** see many uses of Excel

on the last homework. ***Do not use Excel if you are generally using Stata; this assignment is an exception if you do choose to use Excel, but you should generally graph in Stata***.

Syntax is very simple.

```
scatter y x || lfit y x
```

I also generally dislike the legend. Here are some customizations.

```
scatter y x, legend(off) || lfit y x, title("Your title here") ///
    xtitle("{it:X}-axis title here") ytitle("{it:Y}-axis title here") ///
    note("Your source here")
```

3 Appendix

This is all FYI, especially part two.

3.1 Two versions of the "computational formula"

Here's how we get from the formula Prof. Gerber showed to the one I show (the one I'm showing is perhaps more common but not more "correct").

$$
\begin{aligned}
r &= \frac{n\sum x_j y_j - \sum x_j \sum y_j}{\sqrt{n\sum x_j^2 - \left\{\sum x_j\right\}^2}\sqrt{n\sum y_j^2 - \left\{\sum y_j\right\}^2}} \\
&= \frac{n^2\overline{xy} - n\bar{x}n\bar{y}}{\sqrt{n(n\overline{x^2}) - (n\bar{x})^2}\sqrt{n(n\overline{y^2}) - (n\bar{y})^2}} \\
&= \frac{n^2(\overline{xy} - \bar{x}\bar{y})}{\sqrt{n^2\left\{\overline{x^2} - \bar{x}^2\right\}}\sqrt{n^2\left\{\overline{y^2} - \bar{y}^2\right\}}} \\
&= \frac{(\overline{xy} - \bar{x}\bar{y})}{\sqrt{(\overline{x^2} - \bar{x}^2)}\sqrt{(\overline{y^2} - \bar{y}^2)}}
\end{aligned}
$$

3.2 Regression: a geometric derivation

So, here is our task: find the best possible line of the form $Y = \beta_0 + \beta_1 X$ in our sample to describe the data. *We'll generally write the sample estimate of this line as* $y = \widehat{\beta}_0 + \widehat{\beta}_1 x$.[4]

Let's again use an alternate geometry to motivate this problem, making it incredibly easy to solve (I think). Here's what I said last lecture about this *subject space*, which we use to represent our sample data. It bears repeating.

[4] You might also see the *betas* with hats on them, like so: $\widehat{\beta}$. This is more technically correct, but it is kind of hard to look at, and I've already used the tradition of using Roman letters for sample estimates, like s for σ and r for ρ.

Recall that above, we were working in what is called *variable space*: the axes of that two dimensional space each represent a variable. But, recall that we also can picture our variables in what is called *subject space*, where each axis is a person. Recall that this is an especially natural tool if we think about *centering* the original scores; so, for example, the whole vector of scores $\mathbf{y}_c = \mathbf{y} - \bar{\mathbf{y}}$.

Here is a picture of how that looks in just two dimensions. On the variable x person 1 had a (centered) score of 4; person 2 had a score of zero (centered; I won't write that explicitly anymore). So, that variable is the vector $(4, 0)$ (a vector is just a point drawn as an arrow). On variable y, person 1 had a 3, while person 2 had a 3 also, so that is the vector $(3, 3)$.

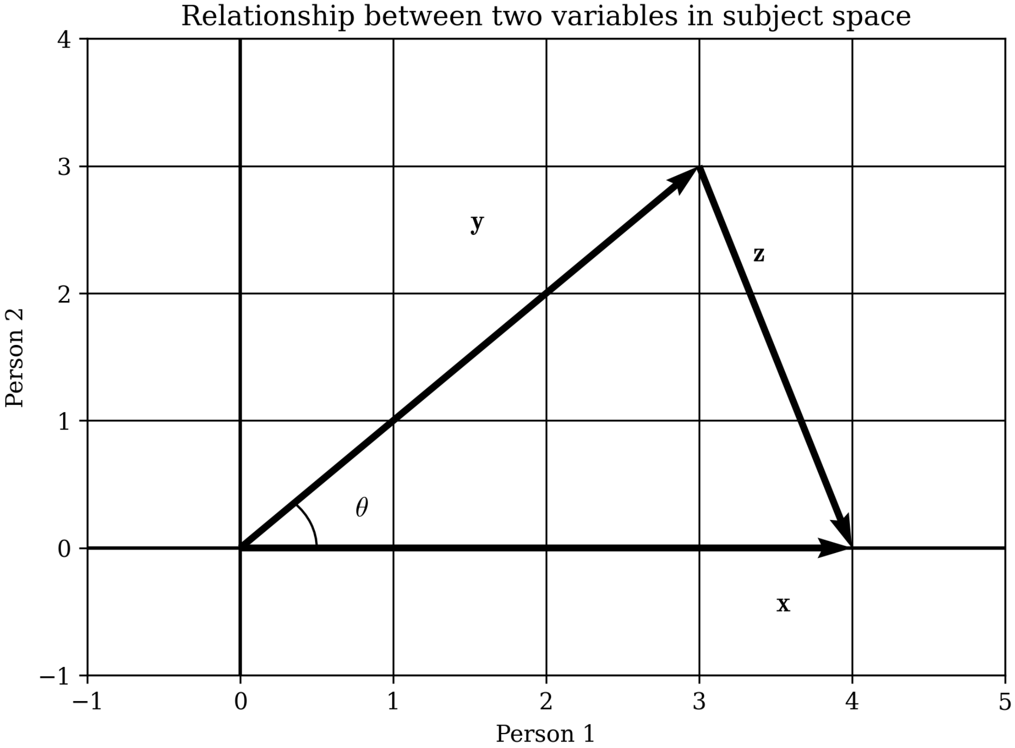

Figure 1: cosines

The idea here is very simple. **I want to find some multiple of x in the sample to get it as close to as possible to y (stretching a vector representing x in subject space by *a constant $\hat{\beta}_1$, the sample estimate of β_1*, to produce a list of predictions $\hat{\mathbf{y}}$ is equivalent to drawing a line $\hat{y} = \hat{\beta}_1 x$ in variable space).**

3.3 Regression slope: basic formula

Now, the generalized law of cosines gives us our answer for how to find the perfect sample estimate $\hat{\beta}_1$ of our population regression slope β_1. We see above that our error vector is

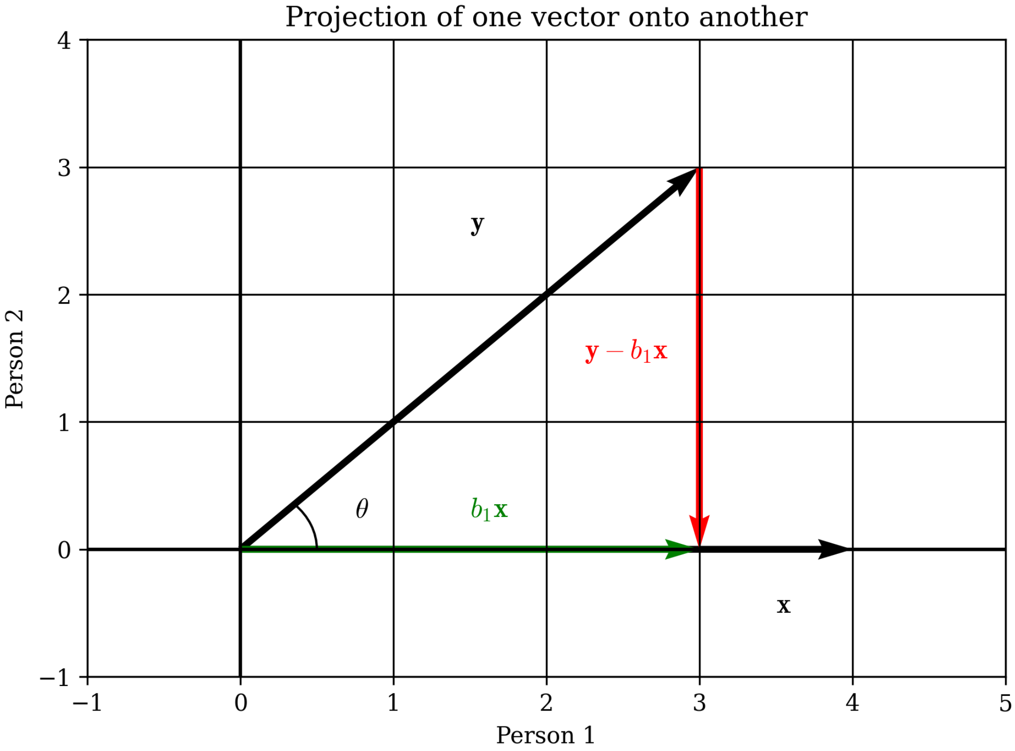

Figure 2: projreg

smallest if it is perpendicular to our predictor. Let's assume for now that our intercept β_0 is zero; we'll motivate this in a moment.

The law of cosines tells us that for two vectors to be perpendicular (or *orthogonal*, a term more common in this context), their "dot product" must be zero.

What is this scary-sounding "dot product"? You'll be very pleased to learn that it is something we have already calculated: for two vectors $\mathbf{w}$ and $\mathbf{z}$, it is $\mathbf{w} \cdot \mathbf{z} = \sum_{j=1}^{n} w_j z_j$.

This looks very much like a correlation's numerator, especially if we consider that these vectors in the picture represent *centered* versions of our variables: then, for our centered error and outcome vectors to be orthogonal, we have $[(\mathbf{y} - \overline{\mathbf{y}}) - \hat{\beta}_1(\mathbf{x} - \overline{\mathbf{x}})] \cdot (\mathbf{y} - \overline{\mathbf{y}}) = \mathbf{0}$, which can be written in summation notation as $\sum_{j=1}^{n}[(y_j - \bar{y}) - \hat{\beta}_1(x_j - \bar{x})](y_j - \bar{y}) = 0$.

So, let's make our residual vector or in-sample error as small as possible by making it orthogonal to our predictor (for the simple reason that the shortest distance between two points is a straight line; our error vector is smallest if it is a straight, not oblique, line from the scaled predictor to the outcome).

First, write both quantities in centered form; for the in-sample error, it is $[(y_j - \bar{y}) - \hat{\beta}_1(x_j - \bar{x})]$.

$$\sum_{j=1}^{n}[(y_j - \bar{y}) - \hat{\beta}_1(x_j - \bar{x})](x_j - \bar{x}) = 0$$

$$\sum_{j=1}^{n}(y_j - \bar{x})(x_j - \bar{x}) = \hat{\beta}_1 \sum_{j=1}^{n}(x_j - \bar{x})^2$$

$$\frac{\sum_{j=1}^{n}(y_j - \bar{y})(x_j - \bar{x})}{\sum_{j=1}^{n}(x_j - \bar{x})^2} = \hat{\beta}_1$$

3.4 Regression slope: close relation to correlation

Even better news! Doesn't this look *very* similar to our correlation coefficient?

$$r = \frac{\sum_{j=1}^{n}(y_j - \bar{y})(x_j - \bar{x})}{\sqrt{\sum_{j=1}^{n}(x_j - \bar{x})^2}\sqrt{\sum_{j=1}^{n}(y_j - \bar{y})^2}}$$

In fact, we can just multiply the correlation coefficient by the ratio $\frac{s_y}{s_x}$ to get $\hat{\beta}_1$. The algebra is easy but tedious. *I encourage you to try this on your own; see appendix for proof.*

3.5 Regression slope: centered example, motivating the intercept

Let's tarry no further. What should the *slope of our line of best fit* for the height/logged weight data above? Recall that we had a correlation of $r = 0.8942$ and the following summary statistics (I'm being a bit more precise than normal here since one of our variables is logged)…

	ht	logwt
mean	156.5864	4.1066
std	22.2551	0.5601

So, what is the slope of our line of best fit? Simply, $\hat{\beta}_1 = r \cdot \frac{s_y}{s_x} = 0.8942 \cdot \frac{0.5601}{22.2551} = 0.0225$.

Now, what should our estimate of the **intercept** ($\hat{\beta}_0$ in the sample; β_0 in the population) be? I can give you a formula here, but what if I motivate it a bit?

Recall that above we had centered data. The *centering* operation is very often useful, as we've seen. It changes little except very basic summary statistics. Then, we would take our population model, $Y = \beta_0 + \beta_1 X_1 + \epsilon$, and rewrite it. To find the model for centered data, we can write $(Y - \mu_Y) = \beta_1(X_1 - \mu_X) + \epsilon$ by simply taking the deviation of both sides from their mean.[5] The intercept drops out because its mean is just itself. We can rewrite our sample regression equation accordingly as simply $(y - \bar{y}) = \hat{\beta}_1(x - \bar{x})$. Then, we have no need for a constant term. When our centered predictor $(x_j - \bar{x}) = 0$, so must y by construction. Our regression line for centered variables looks like this.

So, what if we *don't* have mean-zero variables? Well, our slope remains correct—geometrically, you can think about the process of restoring the means in variable space as just shifting all of our points right and up (assuming both had positive means); this kind of translation can't affect the slope, of course.

Rearranging our population model so that we now state it in terms of *uncentered* variables, we have the following: $Y = \mu_Y + \beta_1 X - \beta_1 \mu_X + \epsilon$. Our sample regression line, accordingly, is $y = \bar{y} + \hat{\beta}_1 x - \hat{\beta}_1 \bar{x}$.

This tells us that, with uncentered variables, our best prediction is the slope from the centered model multiplied by x, but with an extra $\bar{y} - \hat{\beta}_1 \bar{x}$ added on. We call it our **intercept and denote it** $\hat{\beta}_0$ in the sample.

So, generally…

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

Our intercept for this model is therefore $4.1066 - 0.0225 \cdot 156.5864 = 0.58$. So, our regression line now looks like this: $y = 0.58 + 0.0225x$.

3.6 Regression diagnostics

Now that we've worked an example—the actual computations here are very simple if we have the correlation coefficient, standard deviations, and means handy, so I'm punting on more computational practice until the end—let's talk about how to assess our model.

[5] By linearity of expectations, the mean of the right-hand side is just the sum of the means of each variable: ϵ has mean-zero by construction, and β_0 is constant, so it is its own mean and thus subtracts itself out.

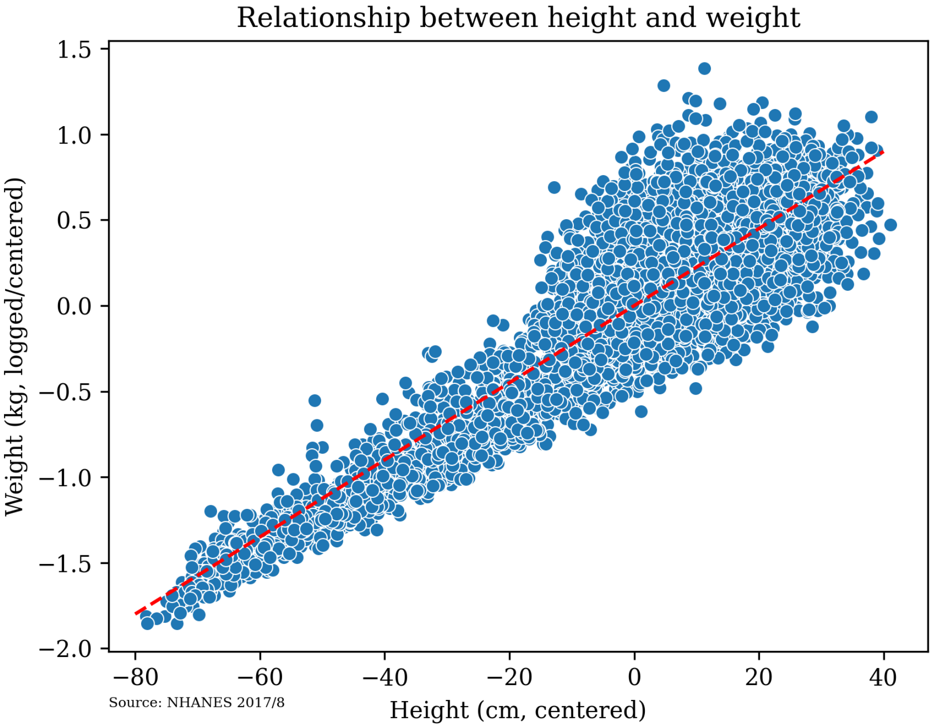
Relationship between height and weight
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
Weight (kg, logged/centered)
−80
−60
−40
−20
0
20
40
Height (cm, centered)
Source: NHANES 2017/8

Figure 3: reg

Recall the two main assumptions of our model.

Firt, we assumed that we we included all relevant variables in the model. Failure to include the correct variables in the model makes our slope wrong in terms of the causal effects of X, but it can still be valid descriptively. We'll see an example later.

Second, we assumed that we had the right functional form. Since we're working here with *linear* model, this means that the actual data-generating process actually work like our model says it does: $Y = \beta_0 + \beta_1 X + \epsilon$. The predictor X itself can be something like a square or logarithm of some underlying variable (indeed, that's what we've already seen), but the model must be linear in its *parameters*: if we have multiple variables, the model is linear if and only if we just multiply the variables by constants and then add to get predictions. *Other types of functional forms are possible, and our linear model is misspecified if reality is not linear like this.*

In what follows, I first address the question of quantifying out model fit—how good are our predictions?—which can be seen as a means of sussing out whether we have the right variables in our model and the right functional form. Then, I discuss head-on some more-direct, intuitive ways of looking at whether our assumptions are correct and what to do if they aren't.

3.6.1 Predictions, errors, the mean squared error, and model fit

Let's talk about model fit. This stuff is actually very interesting, rather than just being a bunch of nitpicking; it's quite geometrically beautiful.

First: how good of a fit is our model? Well, unsurprisingly, we'll again use the mean squared error as a measure. We saw before that conditional means minimized the MSE, and if our model is truly a linear one, our simple formula $b_1 = r \cdot \frac{s_y}{s_x}$ is the regression slope that minimizes in-sample errors.[6]

These in-sample errors, training errors, or residuals are the gap *in the sample* between our prediction, written $\hat{y}_j = b_1 x_j + b_0$, and our actual values, written of course y_j. Note that with linear regression we almost always have points with the same input value x_j so that many different individuals share the same prediction.

It is important to distinguish in-sample errors or residuals (same thing) from the out-of-sample errors, the latter of which combine error arising from the fact that our sample model is not only never-quite-perfect for data we *do* have but also is itself only a guess at the true model parameters (because estimated from sample data).[7] **We will use the "hat" notation $\hat{y}_j$ for our *prediction* for person j.** This is just equal to the model's value at person j's value. Then, **the error for person j is simply** $err_j = y_j - \hat{y}_j$. This is how much we miss at that point.

[6] Making the error vector perpendicular to the predictor also minimizes it; you can formally write this as a minimization problem and solve the regression problem with calculus—but I think the linear algebra approach is even prettier).

[7] Another way to say this is: our sample model is the right model for the "population" that is our sample, assuming our theoretical selection of variables and their functional form is correct. It still won't be perfect, of course. But, for data points not in our sample, we also have to deal with the fact that the model is not the right model for the actual population, just a guess at it. So it will miss points not in our sample for that reason as well.

Now, a global measure of error would simply be the mean squared error, that same loss function that we've used time and again, sometimes as a total, sometimes as a mean, sometimes with the root taken—but always that same sum of squared deviations appears. Pretty handy how this stuff doesn't change that much, huh?

In a regression context, the (near) average of the square deviations is called the "mean squared error" and standard statistical software reports it or its root, the "root mean squared error". These are basically the variance and standard deviation of the data but with the prediction for a person swapped in where we had the mean before (and recall that the mean is our most basic predictor, so it's not even that different). *The sum of squared errors is also the length of that error vector we saw before!*

Just make sure that you remember that this is the *in-sample* error, which is why it is sometimes called a "residual" (to distinguish it clearly from out-of-sample error). *Older books and software are frustratingly inconsistent in this language, so I will be ultra-precise: residual = in-sample error =/= out-of-sample error.* In general, to interpret "error", you need to know the context.

$$\begin{aligned} SS_E &= \sum_{j=1}^{n}(y_j - \hat{y}_j)^2 \\ MSE &= \frac{1}{n-1}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2 \\ RMSE &= \sqrt{MSE} \end{aligned}$$

3.6.2 The "coefficient of determination", aka R-squared

The MSE is a nice quantity to have around, but it, like all variance-forms, is not a quantity we can just compare blindly across contexts. Is there a form of model fit that is perhaps more like a percentage, which can at least sort of be compared? Yes. We can scale the MSE by dividing it by the overall variation, $\frac{MSE}{s_y^2}$; this quantity is then conventionally subtracted from 1 to give it a positive interpretation: **how much of the total variance in the outcome does our model explain?** Logically, the two must sum to 1, another of our probability/counting rules. This also has a very pleasing geometric interpretation.

First, let's just write out this ratio, whose name is R-squared or R^2. You might wonder if this is related to the correlation coefficient. It is! (Why it is not just written r^2, I don't know). In what follows, remember that the numerator of the variance, $\sum_{j=1}^{n}(y_j - \bar{y})^2$, is often referred to as the *total sum of squares*, or SS_T. *Recall that this is just the squared length of a vector if we draw a centered version of the data as a vector.*

$$\begin{aligned}
R^2 &= 1 - \frac{MSE}{s_y^2} \\
&= 1 - \frac{\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}{\sum_{j=1}^{n}(y_j - \bar{y})^2} \\
&= 1 - \frac{\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}{SS_T} \\
&= 1 - \frac{SS_E}{SS_T}
\end{aligned}$$

The final line has a very nice interpretation in the centered subject-space picture we've been using off and on throughout all the notes: it is just one minues the ratio of the squared length of the vector representing our in-sample errors to the squared length of the vector representing our variable. Pretty easy to interpret, no? Statistical software reports these quantities when you carry out a regression.

Even better, consider the fact that since the in-sample error vector is totally orthogonal (perpendicular) to the vector representing our predictor.[8] This means that by the Pythagorean theorem, we can immediately calculate R^2 by simply finding the the squared length of that prediction vector, conventionally called the "model sum of squares" (SS_M), and dividing it by the total squared length of our outcome to get our model fit.

The squared length of our prediction vector is called the "model sum of squares" and written SS_M. The formula for it is as follows:

$$SS_M = \sum_{j=1}^{n}(\hat{y}_j - \bar{y})^2$$

Then, **we can say that by the Pythagorean theorem, our quantity R^2 is just the ratio of our prediction vector's squared length to the total sum of squares! So, we can also write $R^2 = 1 - \frac{SS_E}{SS_T} = \frac{SS_T - SS_E}{SS_T} = \frac{SS_M}{SS_T}$.**

By the way, for the exact way that these formulae for sums of squares connect to the geometry, see the appendix.

3.6.2.1 Correlation connection So, do we have to calculate this beastly thing in order to find our model fit? Actually, no. Believe it or not, there is a major shortcut!

Notice that the ratio of the squared length of our predictors to the total squared length of $\mathbf{y}$ is just the ratio of the adjacent side of the angle between them, $\theta_{\mathbf{x},\mathbf{y}}$, to the hypotenuse of a right triangle. That means that this ratio is … the square of $\cos(\theta)$! If you recall from last lecture that the correlation coefficient is the cosine of θ, we can then just take the square of the correlation coefficient and obtain our model fit. **Hence the name "R-squared".**

[8] And also the predictions, since they are just multiples of one another: $\hat{y}_j = b_1 x_j$ for all j.